Ömer Tuğrul
Section - 1
22002723

CS464: Introduction to Machine Learning - Homework 2

Table of Contents

Ömer Tuğrul
Section - 1
22002723

PART 1-PCA:

In the first part of the assignment, I preprocessed the images and flattened them to be 784-dimensional vector for each image.

```
images.shape
```

```
(60000, 784)
```

```
labels.shape
```

```
(60000,)
```

Principal component analysis algorithm:

I normalized the pixels before doing PCA.

1- Center the data

2- Compute the covariance matrix with the equation below:

$Cov = \frac{(X^T X)}{N}$, where N is the number of rows and X is image matrix.

3- Do eigenvalue decomposition to find the eigenvalue and eigenvectors.

4- Order the eigenvalues in descending order, the eigenvector corresponding to highest eigenvalue will show the direction which captures the most variance.

5- First eigenvector will be first principal component and it goes like this.

Q1.1

```
PVE for first 10 principical components are 0.48814982
```

According to this result, the proportion of variance explained by the first 10 principal components is approximately 49%. If we only use these components, we will lose significant amount of variance in our data, and this can lead to false predictions.

Principal Component	PVE
1	0.09704664
2	0.07095924
3	0.06169089
4	0.0538942
5	0.04868797
6	0.04312231
7	0.0327193
8	0.02883895
9	0.02762029
10	0.02357001

Ömer Tuğrul
Section - 1
22002723

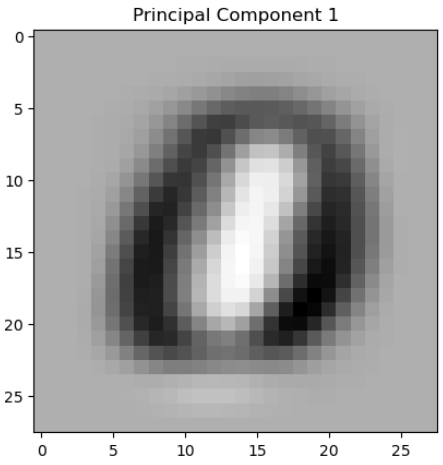
Q1.2

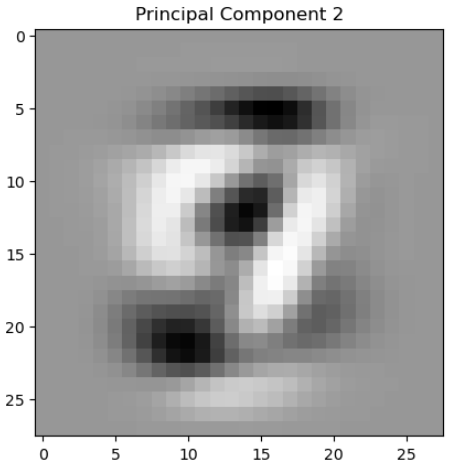
```
sum_total_eig = np.sum(eig_val)
pve = 0
count = 0
while pve <= 0.7:
    pve += eig_val[count] / sum_total_eig
    count +=1
print("At least",count,"number of components need to explain 70% of varience.")
```

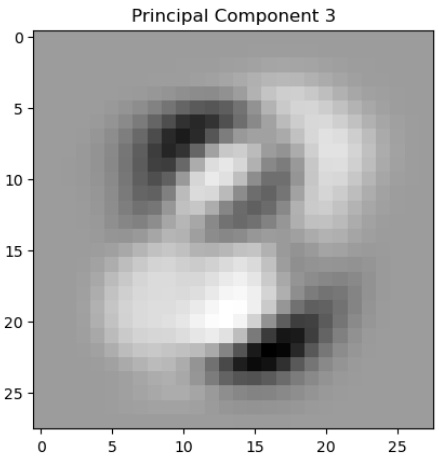
```
At least 26 number of components need to explain 70% of varience.
```

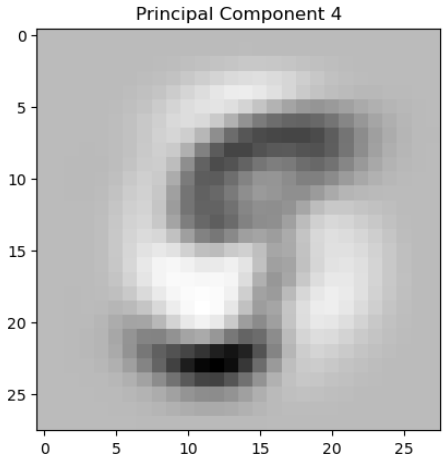
We need **at least 26 number of principal components** to capture the 70% of variance.

Q1.3

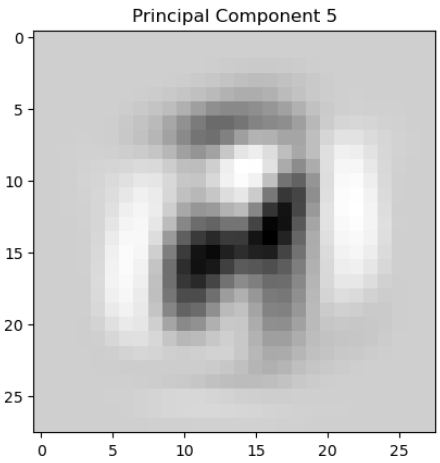


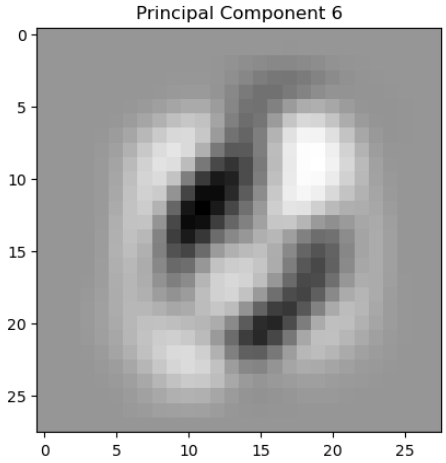


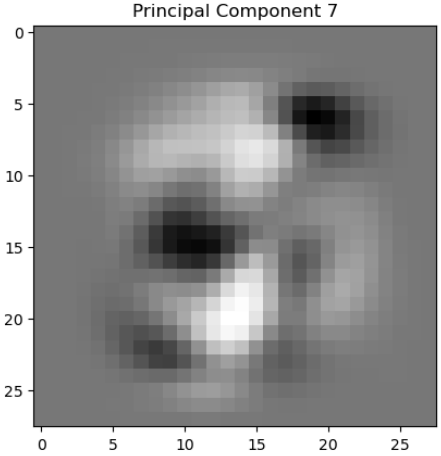


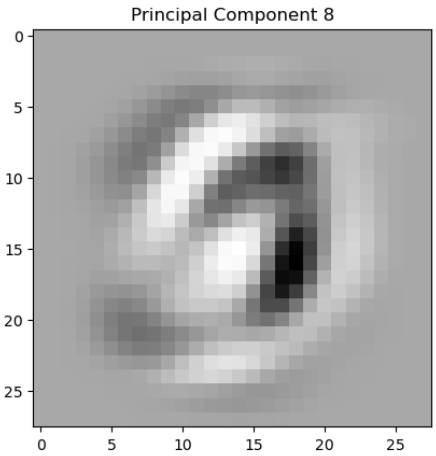


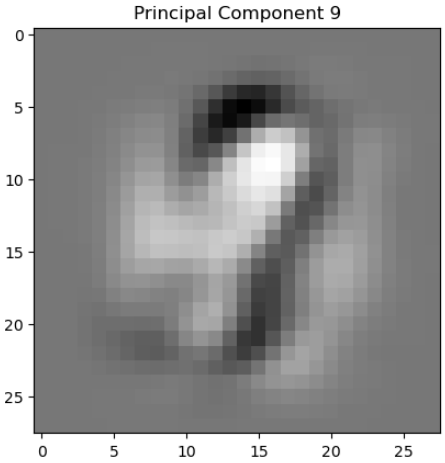
Ömer Tuğrul
Section - 1
22002723

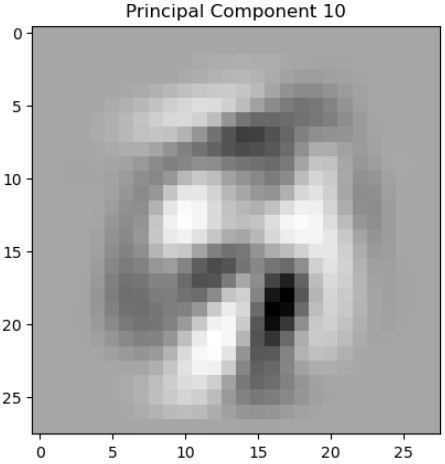








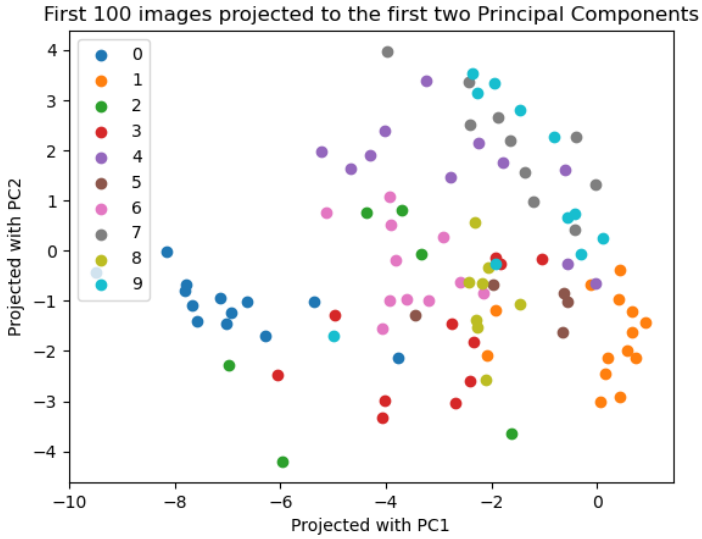




In the above, the first 10 principal components are displayed. Each principal captures certain variance(features) of the data. The first principal components capture the most variance in the data. These components are used in dimensionality reduction. By projecting the high-dimensional data onto those components, one can reduce the number of features while retaining vast amount of variance.

Ömer Tuğrul
Section - 1
22002723

Q1.4



We can see from the above plot that some digits like “0” and “1” are quite distinct from the others. This suggests that certain digits can be distinguished more clearly than others based on properties captured by the first two major components. Strong cluster separation indicates that the principal components are able to capture the distinctive characteristics of those digits. For instance, the number '0' is clearly distinguished from other numbers, suggesting that it possesses special characteristics of its own. Clusters that overlap indicate that the associated digits have characteristics in common that the first two principal components do not distinguish.

Q1.5

In this part, the first image is reconstructed with different number of principal components. The algorithm for reconstruction of the image is as following:

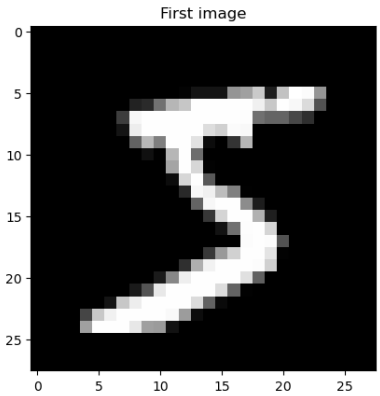
1- Center the original image.

2- Project the centered image onto principal components.

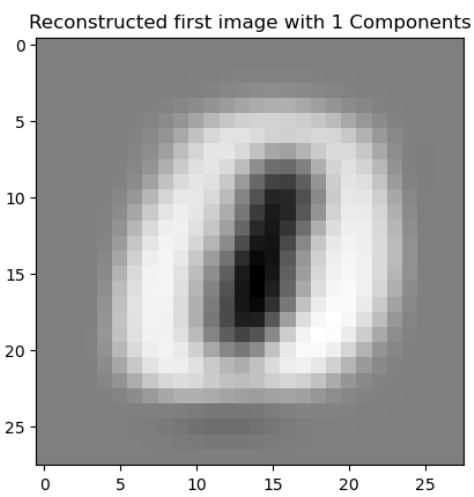
$$z = X_{centered} \cdot PC_k$$ (Where PC_k is first k principal components.)

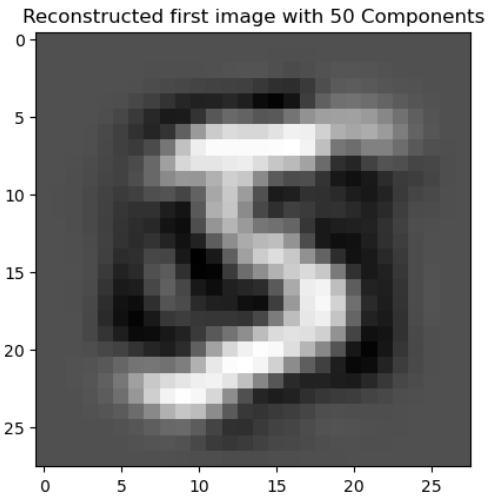
3- Reconstruct the image from the principal component coefficients.

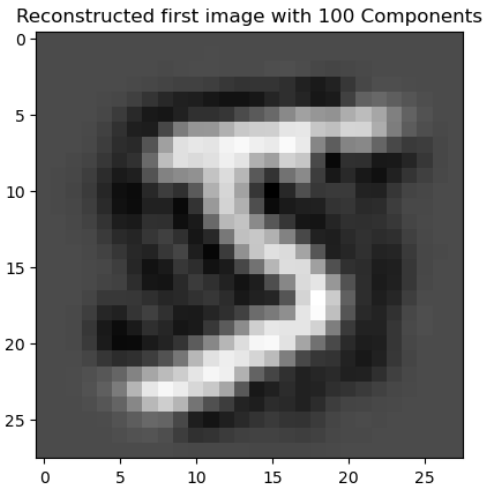
$$v = z \cdot {PC_k}^T + \mu$$ (Where v is the reconstructed image vector and μ is mean.)

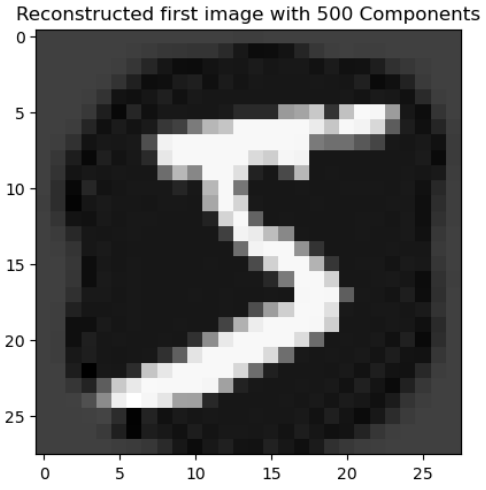


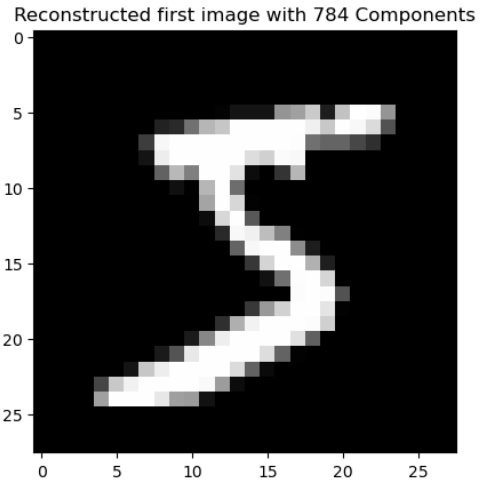
Ömer Tuğrul
Section - 1
22002723











When k is small, it is seen that reconstruction will capture the most important features of the images, but it loses many details. As the number of principal components increases, the quality of the reconstruction increases, and the reconstruction becomes more accurate. When there are 784

principal components used in the reconstruction, the reconstructed images is identical to the original image.

PART 2- Multinomial Logistic Regression:

The hand-written digits are size-normalized and centered. Then, to obtain the validation set 10000 images from the training set separated to validation.

```
X_train.shape,X_val.shape,X_test.shape
```

```
((50000, 784), (10000, 784), (10000, 784))
```

Furthermore, to implement the multinomial logistic regression classifier, the labels are converted to the one-hot-encoded form to predict 10 different classes.

```
y_train.shape,y_val.shape,y_test.shape
```

```
((50000, 10), (10000, 10), (10000, 10))
```

Q2.1 – Base Model Results

For the base model, the hyperparameter are configured as the following:

- Regularization coefficient = 10^{-4}
- Learning rate = $5 \cdot 10^{-4}$
- Batch size = 200
- Number of epochs = 100
- İnitialization of weights = Gaussian distribution

Results:

```
Accuracy score for default model is 0.907
```

Confusion matrix for the base model:

	Pred 0	Pred 1	Pred 2	Pred 3	Pred 4	Pred 5	Pred 6	Pred 7	Pred 8	Pred 9
True 0	944	0	1	3	1	15	10	3	2	1
True 1	0	1095	7	3	0	5	4	1	20	0
True 2	3	10	911	20	10	8	14	10	36	10
True 3	3	1	23	904	0	33	3	13	25	5
True 4	1	2	8	3	893	2	13	5	10	45
True 5	9	5	11	36	12	766	16	6	24	7
True 6	9	3	9	2	14	22	893	4	2	0
True 7	0	8	23	13	10	2	0	927	2	43
True 8	6	6	14	29	11	29	14	8	840	17
True 9	8	7	2	9	38	6	1	26	15	897

Other performance metrics for the base model:

Ömer Tuğrul
Section - 1
22002723

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
Precision	0.942517	0.959930	0.912831	0.882066	0.901408	0.852349	0.932984	0.932731	0.881847	0.874269
Recall	0.970408	0.970925	0.903101	0.896040	0.912424	0.854260	0.930063	0.903696	0.842916	0.888999
F1 Score	0.956259	0.965396	0.907940	0.888998	0.906883	0.853303	0.931521	0.917984	0.861942	0.881572
F2 Score	0.964699	0.968706	0.905030	0.893210	0.910199	0.853877	0.930645	0.909358	0.850425	0.886013

Q2.2 – Hyperparameter selection

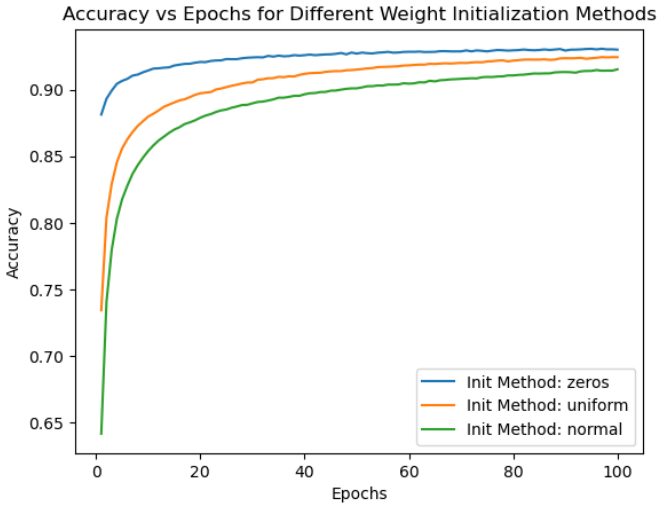
For the hyperparameter selection, four different experiments are conducted. The hyperparameters will change one at a time and the others will remain constant. Hyperparameters given in the task are given as the following:

- Batch size: 1, 64, 50000
- Weight initialization technique: zero initialization, uniform distribution, normal distribution
- Learning rate: 0.1, 10^{-3}, 10^{-4}, 10^{-5}
- Regularization coefficient (λ): 10^{-2}, 10^{-4}, 10^{-9}

Q2.2.1- Different Weight Initializations

For the first experiment, the hyperparameters other than the weight initialization method is fixed as the following:

- Regularization coefficient = 10^{-4}
- Learning rate = $5 \cdot 10^{-4}$
- Batch size = 64
- Number of epochs = 100

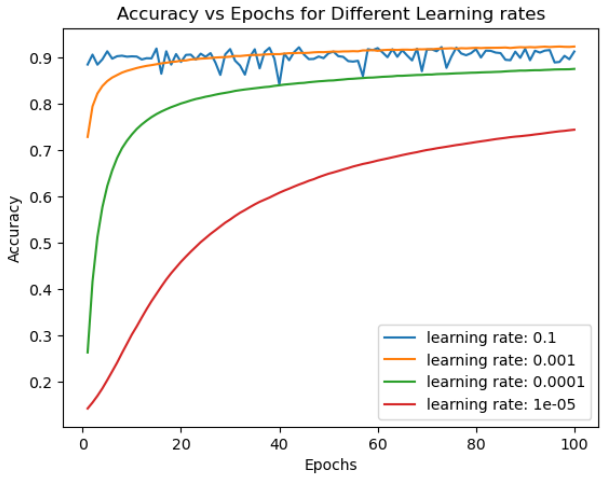


Q2.2.2- Different Learning Rates

For the second experiment, the hyperparameters other than the learning rate is fixed as the following:

- Regularization coefficient = 10^{-4}
- Batch size = 64
- Number of epochs = 100

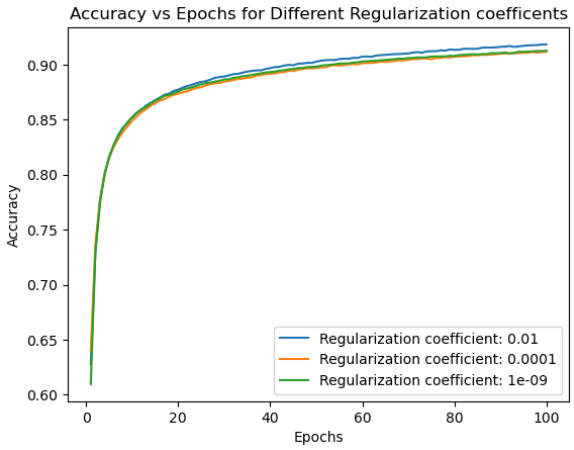
- Weight initialization method = Gaussian distribution



Q2.2.3- Different Regularization Coefficients

For the third experiment, the hyperparameters other than the regularization coefficient is fixed as the following:

- Learning rate = $5 \cdot 10^{-4}$
- Batch size = 64
- Number of epochs = 100
- Weight initialization method = Gaussian distribution

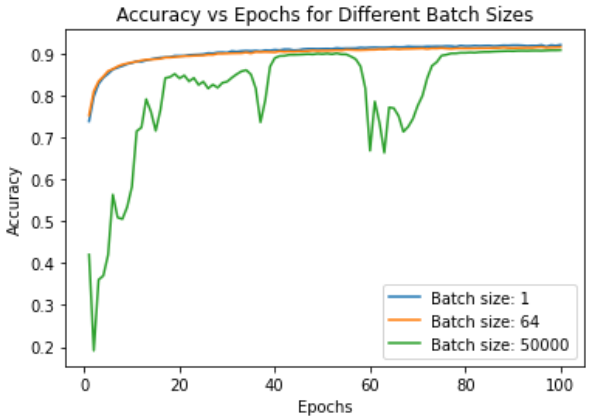


Q2.2.3- Different Batch Sizes

For the fourth experiment, the hyperparameters other than the batch size is fixed as the following:

- Learning rate = $5 \cdot 10^{-4}$
- Regularization coefficient = 10^{-4}

- Number of epochs = 100
- Weight initialization method = Gaussian distribution



Q2.3 – Choosing the Best Model

After the experiments above, the hyperparameters and weight initialization method for the best model is selected as the following:

- Regularization coefficient = 10^{-2}
- Learning rate = 10^{-3}
- Batch size = 64
- Number of epochs = 100
- İnitialization of weights = Zeros

Results:

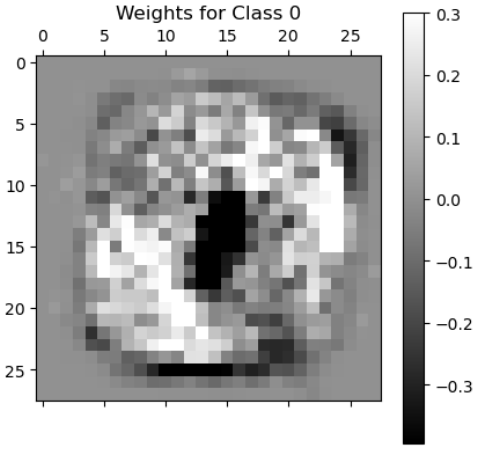
```
Accuracy score for best model is 0.9255
```

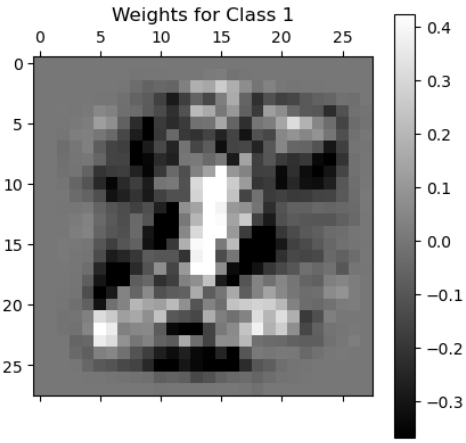
Confusion matrix:

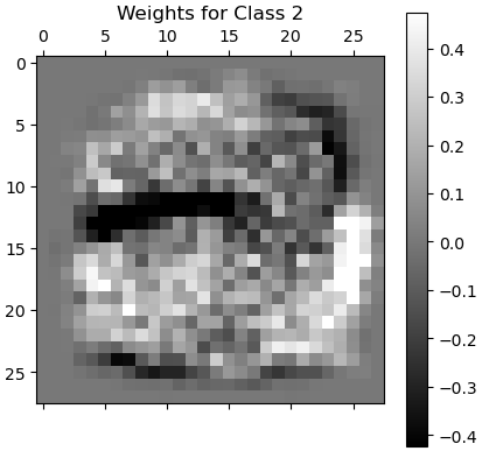
	Pred 0	Pred 1	Pred 2	Pred 3	Pred 4	Pred 5	Pred 6	Pred 7	Pred 8	Pred 9
True 0	963	0	1	3	0	4	6	2	1	0
True 1	0	1111	2	2	0	1	4	2	13	0
True 2	7	8	933	15	8	2	14	9	29	7
True 3	4	0	19	926	0	19	2	11	20	9
True 4	1	3	5	1	915	0	10	4	7	36
True 5	10	2	4	40	11	762	18	7	30	8
True 6	12	3	5	2	8	12	913	2	1	0
True 7	1	8	23	7	8	1	0	944	1	35
True 8	9	9	6	27	9	24	11	8	859	12
True 9	11	8	1	9	25	4	0	14	8	929

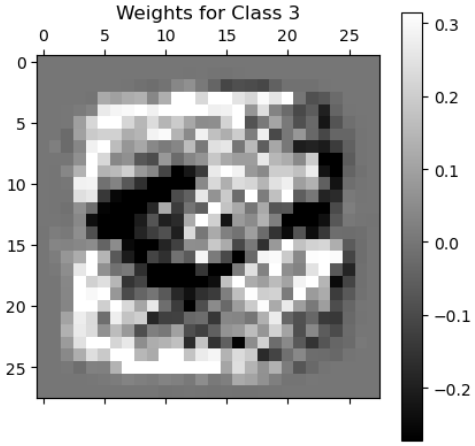
Ömer Tuğrul
Section - 1
22002723

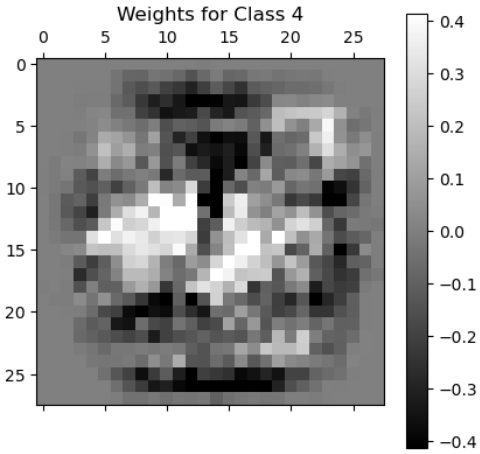
Q2.4 – Visualization of the Final Weights

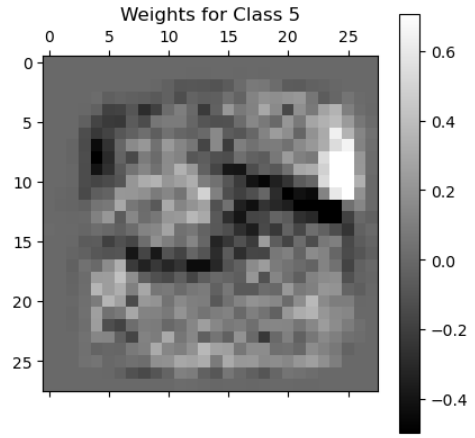




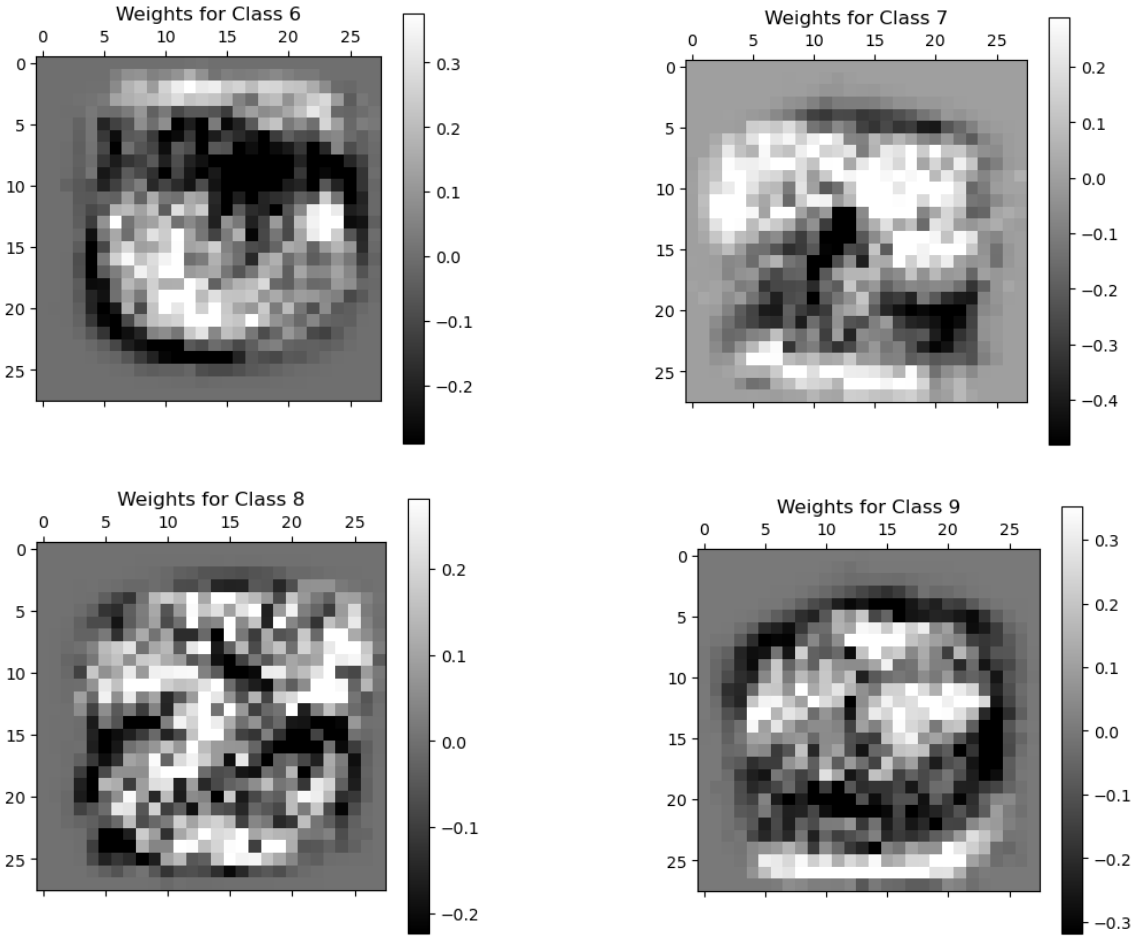








Ömer Tuğrul
Section - 1
22002723



This visualization of the weight vectors represents the importance of each pixel determining the class of an image. For example, for the class 0 the weight vector image shows a dark region in the center, which is typical for 0. For the visualization of weight image for 1, it shows vertical alignment, which is typical for 1. We can go over all classes like this to analyze the weight vector images, and this visualization shows us how model learned to represent different classes by giving different importance to the pixels.

Q2.5 – Performance Metrics for the Best Model

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
Precision	0.945972	0.964410	0.933934	0.897287	0.929878	0.919180	0.933538	0.941176	0.886481	0.896718
Recall	0.982653	0.978855	0.904070	0.916832	0.931772	0.854260	0.953027	0.918288	0.881930	0.920714
F1 Score	0.963964	0.971578	0.918759	0.906954	0.930824	0.885532	0.943182	0.929591	0.884200	0.908557
F2 Score	0.975091	0.975931	0.909889	0.912855	0.931393	0.866500	0.949064	0.922776	0.882837	0.915812

A complete picture of the model's performance is provided by the confusion matrix given in Q2.3 and the weight images given in the Q2.4. The confusion matrix statistically displays the frequency with which one class is confused with others, while the weight images provide information about the attributes that are being identified for each class. When we look at the confusion matrix, it is seen that some classes are more prone to misclassification than others. For example, class 5 is more prone to misclassification than class 1. We can understand why it is the case by looking at the weight vector

images of different classes. For instance, for the classes 0 and 1 weight vector images shows clear and distinct patterns. Whereas, for the classes such as 3, 5 and 8 have more overlap in the weight vector images and we see more misclassification about them in the confusion matrix.